

.REM >

IDENTIFICATION

PRODUCT CODE: AC-E715I-MC
PRODUCT NAME: CXDQAI0 DQ11 DEC/X11 MODULE
DATE: SEPTEMBER 1978
MAINTAINER: DEC/X11 SUPPORT GROUP

THE INFORMATION IN THIS DOCUMENT IS SUBJECT TO CHANGE WITHOUT NOTICE AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT CORPORATION. DIGITAL EQUIPMENT CORPORATION ASSUMES NO RESPONSIBILITY FOR ANY ERRORS THAT MAY APPEAR IN THIS MANUAL.

THE SOFTWARE DESCRIBED IN THIS DOCUMENT IS FURNISHED TO THE PURCHASER UNDER A LICENSE FOR USE ON A SINGLE COMPUTER SYSTEM AND CAN BE COPIED (WITH INCLUSION OF DIGITALS COPYRIGHT NOTICE) ONLY FOR USE IN SUCH SYSTEM, EXCEPT AS MAY OTHERWISE BE PROVIDED IN WRITING BY DIGITAL.

DIGITAL EQUIPMENT CORPORATION ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS SOFTWARE ON EQUIPMENT THAT IS NOT SUPPLIED BY DIGITAL.

COPYRIGHT (C) 1973,1978 DIGITAL EQUIPMENT CORPORATION

1. ABSTRACT

DQA IS AN IDMOD THAT WILL EXERCISE UP TO EIGHT DQ11S. DURING A SINGLE PASS IT WILL DO 15 CHARACTER TRANSFERS OF SEQUENTIAL DATA AND CHECK THE STATUS REGISTERS AND SECONDARY REGISTERS. IT WILL TRANSFER A 256 EIGHT BIT BINARY COUNT PATTERN 128 TIMES FOR EACH END PASS RECORDED. ANY ERRORS DETECTED DURING THE PASS ARE REPORTED ON THE CONSOLE TTY.

2. REQUIREMENTS

HARDWARE: ONLY THE BASIC UNIT IS EXERCISED; SO A BASIC UNIT IS NEEDED. ANY EXTRA OPTIONS ON THE DQ11 WILL NOT BE EXERCISED.

STORAGE:: DQA REQUIRES:
1. DECIMAL WORDS: 768
2. OCTAL WORDS: 1400
3. OCTAL BYTES: 3000

3. PASS DEFINITION

ONE PASS OF THE DQA MODULE CONSISTS OF 24576 CYCLES OF AN INCREMENTAL DATA PATTERN TRANSFERRED AT 15. CHARACTER BURSTS.

4. EXECUTION TIME.

RUNNING ALONE ON AN 11/20 ONE PASS TAKES APPROXIMATELY 20 SECONDS. NOTE: PASS TIME IS DEPENDENT UPON BAUD RATE.

5. CONFIGURATION REQUIREMENTS.

DEFAULT PARAMETERS:

DEVADR: 1 VECTOR: 1 BR1: 5 BR2: 5 DEVCNT: 1
USER MUST SPECIFY THE ADDRESS AND VECTOR OF THE FIRST DQ11 AT CONFIGURATION TIME.

6. DEVICE/OPTION SETUP

NO SPECIAL SET NECESSARY. (BASIC UNIT TESTED ONLY)

7. MODULE DESCRIPTION

- A. TESTS FOR THE AVAILABILITY OF UP TO EIGHT DQ11'S
- B. INITIALIZES ALL DQ11'S. SETS ACCORDINLY.
- C. SETS ALL GO BITS AND LEAVES MODULE
- D. GETS TRANS. INTERRUPTS. GETS ALL RECV. INTERRUPTS.
- E. CHECKS ALL STATUS REGISTERS AND SECONDARY REGISTERS
- F. CHECKS ALL DATA. REPORTS ANY ERROR FOUND.
- G. PREPARES DATA TO TRANSMITTED AGAIN.
- H. REPEATS A THROUGH G 128 X 256 CHARS.
- J. REPORTS END PASS AND CONTINUES AS ABOVE.

8. OPERATOR OPTIONS

- A. MODULE LOCATION DVID1 MAY BE CHANGED TO EXERCISE ANY COMBINATION OF DQ11S. BIT 0=DQ110
BIT 1=DQ11 1..... BIT 7=DQ11 7.
- B. IF DVID1=0 AT RUN TIME NO DQ11S WILL BE EXERCISED.

9. NON STANDARD PRINTOUTS

NONE: ALL PRINTOUTS HAVE THE STANDARD FORMAT.
IF YOU NEED HELP IN RUNNING MODULE REFER TO DEC/X11 DOCUMENT.

>

```
129          .LIST      SEQ,LOC,BIN
129          IOMOD <DQAI>,1,1,5,5,7100,31
130          000000"      MODULE 140000,6QAI,1,1,5,5,7100,31
131          000000"      .TITLE  DQAI DEC/X11 SYSTEM EXERCISER MODULE
132          .DXCOM  VERSION 6      23-MAY-78
133          .LIST      BIN
134          *****
135          000000"      BEGIN:
136          000000" 050504 044501 040 MODNAM: .ASCII /DQAI / ;MODULE NAME
137          000005" 000      XFLAG: .BYTE OPEN      ;USED TO KEEP TRACK OF WBUFF USAGE
138          000006" 000001      ADDR: 1+0      ;1ST DEVICE ADDR.
139          000010" 000001      VECTOR: 1+0      ;1ST DEVICE VECTOR.
140          000012" 240      BR1: .BYTE PRTYS+0      ;1ST BR LEVEL.
141          000013" 240      BR2: .BYTE PRTYS+0      ;2ND BR LEVEL.
142          000014" 000001      DVID1: +1      ;DEVICE INDICATOR 1.
143          000016" 000000      SR1: OPEN      ;SWITCH REGISTER 1
144          000020" 000000      SR2: OPEN      ;SWITCH REGISTER 2
145          000022" 000000      SR3: OPEN      ;SWITCH REGISTER 3
146          000024" 000000      SR4: OPEN      ;SWITCH REGISTER 4
147          *****
148          000026" 140000      STAT: 140000      ;STATUS WORD
149          000030" 000224"      IMT: START      ;MODULE START ADDR.
150          000032" 000224"      SPOINT: MODSP      ;MODULE STACK POINTER.
151          000034" 000000      PASCNT: 0      ;PASS COUNTER.
152          000036" 007100      ICNT: 7100      ;# OF ITERATIONS PER PASS=7100
153          000040" 000000      SOFCNT: 0      ;LOC TO COUNT ITERATIONS
154          000042" 000000      HRDCNT: 0      ;LOC TO SAVE TOTAL SOFT ERRORS
155          000044" 000000      SOFPAS: 0      ;LOC TO SAVE TOTAL HARD ERRORS
156          000046" 000000      HRDPAS: 0      ;LOC TO SAVE SOFT ERRORS PER PASS
157          000050" 000000      SYSCNT: 0      ;LOC TO SAVE HARD ERRORS PER PASS
158          000052" 000000      RANUM: 0      ;# OF SYS ERRORS ACCUMULATED
159          000054" 000000      CONFIG: 0      ;HOLDS RANDOM # WHEN RAND MACRO IS CALLED
160          000056" 000000      RES1: 0      ;RESERVED FOR MONITOR USE
161          000058" 000000      RES2: 0      ;RESERVED FOR MONITOR USE
162          000060" 000000      SVR0: OPEN      ;LOC TO SAVE R0.
163          000062" 000000      SVR1: OPEN      ;LOC TO SAVE R1.
164          000064" 000000      SVR2: OPEN      ;LOC TO SAVE R2.
165          000066" 000000      SVR3: OPEN      ;LOC TO SAVE R3.
166          000070" 000000      SVR4: OPEN      ;LOC TO SAVE R4.
167          000072" 000000      SVR5: OPEN      ;LOC TO SAVE R5.
168          000074" 000000      SVR6: OPEN      ;LOC TO SAVE R6.
169          000100" 000000      CSRA: OPEN      ;ADDR OF CURRENT CSR.
170          000102" 000000      SBAADR: OPEN      ;ADDR OF GOOD DATA, OR
171          000104" 000000      ACSR: OPEN      ;CONTENTS OF CSR.
172          000106" 000000      WBAADR: OPEN      ;ADDR OF BAD DATA, OR
173          000108" 000000      ASTAT: OPEN      ;STATUS REG CONTENTS.
174          000110" 000000      ERRTP: OPEN      ;TYPE OF ERROR
175          000112" 000000      ASD: OPEN      ;EXPECTED DATA.
176          000114" 000000      AWD: OPEN      ;ACTUAL DATA.
177          000116" 000000      RSTRT: RSTRT      ;RESTART ADDRESS AFTER END OF PASS
178          000118" 000000      WDMEM: OPEN      ;WORDS TO MEMORY PER ITERATION
179          000120" 000000      WDR: OPEN      ;WORDS FROM MEMORY PER ITERATION
180          000122" 000000      INTR: OPEN      ;# OF INTERRUPTS PER ITERATION
181          000124" 000031      IDNUM: 31      ;MODULE IDENTIFICATION NUMBER=31
182          .REPT      SPSIZ      ;MODULE STACK STARTS HERE.
183          000040
```

```
184          .NLIST
185          .WORD      0
186          .LIST
187          .ENDR
188          000224"      MODSP:
189          *****
```

190 000224* 012767 000010 177662 START: MOV #9,WDTO ;8 WORDS TO MEM PER ITERATION
191 000232* 012767 000010 177656 MOV #-,WDR ;8 WORDS FROM MEM PER ITERATION
192 000240* 012767 000002 177652 MOV #2,INTR ;2 INTERRUPTS PER ITERATION
193 000248* 012767 000002 177542 MOV #2,INTR ;2 INTERRUPTS PER ITERATION
194 000252* 001002 177542 MOV #2,INTR ;2 INTERRUPTS PER ITERATION
195 000254* 104410 000000* ENDS,BEGIN ;END OF MODULE
196 000260* 006200 1\$: ASR R0 ;KILL 1ST DEV-ALREADY COUNTED
197 000262* 103376 BCS 15 ;LOOP TILL FOUND
198 000264* 103376 2\$: ASR R0 ;SHIFT IN NEXT BIT
199 000266* 103011 BCC 35 ;RR IF NO DEVICE HERE
200 000270* 066767 000010 177616 ADD #,WDTO ;DOUBLE WORDS
201 000272* 066767 000010 177612 ADD #-,WDR ;DOUBLE WORDS FROM
202 000304* 066767 000002 177606 ADD #,INTR ;DOUBLE INTERRUPTS
203 000312* 005700 3\$: TST R0 ;ANY MORE DEVICES?
204 000314* 001363 BNE 25 ;RR IF YES
205 000316* 016767 177472 002432 MOV #DVID1,SELECT ;COPY THE DEVICE SELECTION PARAMETER
206 BR DVID1,SELECT ;INTO SELECT
207 000324* 000402 BR RESTRT ;IF ZERO, NO DEVICES SELECTED-
208 DROP: ENDS,BEGIN ;DROP THE MODULE
209 000326* 104410 000000* RESTRT: CLR# SELECT+1 ;ELIMINATE IRRELEVANT BITS
210 000332* 105067 002421 SETUP1: MOV SELECT,R1 ;COPY SELECT INTO R1 FOR SETUP
211 000334* 001771 BQ DROP ;IF SELECT WAS ZERO, GO DROP THE MODULE
212 000336* 016700 177440 MOV VECTOR,R0 ;IF SELECT WAS NOT ZERO, DO MODULE PROCESSING
213 000338* 016702 177432 MOV ADDR,R2 ;LOAD R0 WITH FIRST VECTOR ADDRESS
214 000340* 012703 002174* MOV #LNKTAB,R3 ;POINT R3 TO BEGINNING OF JSR LINK TABLE
215 000342* 006201 1\$: ASR R1 ;ISOLATE A LINE IN THE "CM" BIT
216 000344* 001427 BQ SETUP2 ;IF NO MORE TO BE SELECTED
217 000346* 062700 000010 ADD #10,R0 ;IF MORE UPDATE ALL REGISTERS..THE VECTOR
218 000348* 062703 000016 ADD #16,R3 ;AND THE LN* TABLE POINTER
219 000350* 000766 BR 15 ;CHECK TO SEE IF OTHER DEVICES SELECTED
220 000352* 010320 2\$: MOV #R3,(R0)+ ;LOAD THE RECEIVING ROUTINE
221 000354* 116720 177401 MOV#B R2,(R0)+ ;POINT R IN THE PROPER VECTOR
222 000356* 005200 INC R0 ;LOAD THE RECEIVER BUFFER
223 000358* 022323 CMP #R3,(R3)+ ;UPDATE R0 TO NEXT VECTOR BOUNDARY
224 000360* 010323 MOV #R3,(R3)+ ;UPDATE R3 TO THE CSR INSERT LOCATION
225 000362* 022323 CMP #R3,(R3)+ ;LOAD THE TRANSMIT CSR INTO LINKING TABLE
226 000364* 010320 MOV #R3,(R2)+ ;UPDATE THE POINTER TO NEXT INSTRUCTION AND
227 000366* 010320 MOV #R3,(R2)+ ;POINT R2 TO THE TRANSMITTER CSR
228 000368* 010320 MOV #R3,(R2)+ ;LOAD THE TRANSMITTER ROUTINE
229 000370* 116720 177362 MOV#B R1,(R0)+ ;POINT R IN THE PROPER VECTOR
230 000372* 005200 INC R0 ;LOAD THE TRANSMIT BUFFER
231 000374* 022323 CMP #R3,(R3)+ ;UPDATE THE VECTOR POINTER
232 000376* 022323 MOV #R3,(R3)+ ;UPDATE THE LINK TABLE POINTER
233 000378* 022323 MOV #R3,(R3)+ ;LOAD THE TRANSMIT CSR INTO LINKING TABLE
234 000380* 062702 ADD #R2 ;UPDATE THE ADDRESS POINTER
235 000382* 000746 BR 15 ;GO SET UP NEXT DEVICE

246 000444* 012767 002646* 001734 SETUP2: MOV #XMTQUE,XMTQPI ;SET UP TRANSMIT QUE ENTRY(IN)
247 000446* 012767 002646* 001730 MOV #XMTQUE,XMTQPO ;POINT
248 000448* 012767 002646* 001730 MOV #XMTQUE,XMTQPO ;SET UP TRANSMIT QUE RETRIEVAL
249 000450* 012767 002646* 001730 MOV #XMTQUE,XMTQPO ;POINT
250 000452* 012767 002666* 001724 MOV #RCVQUE,RCVQPI ;SET UP RECEIVER QUE POINTERS
251 000454* 012767 002666* 001720 MOV #RCVQUE,RCVQPO ;POINT
252 000456* 012767 002706* 001714 MOV #ERRQUE,ERRQPI ;SET UP ERROR QUE POINTERS
253 000458* 012767 002706* 001710 MOV #ERRQUE,ERRQPO ;POINT
254 000460* 012704 000154 MOV #R0,R4 ;LOAD R4 WITH NUMBER OF PUFFER
255 000462* 012703 002426* MOV #XMTBUF,R3 ;LOCATIONS TO BE CLEARED
256 000464* 005023 1\$: CLR (R3)+ ;BEGIN CLEARING BUFFERS AT THE
257 000466* 005304 DEC R4 ;ZERO EACH LOCATION AND POINT R3 TO
258 000468* 001375 BNE 15 ;NEXT WORD
259 000470* 005304 2\$: DEC R4 ;HAVE ALL LOCATIONS BEEN CLEARED?
260 000472* 001375 BNE 15 ;IF NO, CLEAR THE NEXT ONE
261 000474* 016700 002224 ACTIVATE: MOV SELECT,R0 ;LOAD DEVICE SELECTION PARAMETER
262 000476* 110067 002240 MOV R0,DMFLG ;SETUP TRANSFER COMPLETION FLAG
263 000478* 016701 172444 MOV ADDR,R1 ;COPY THE BASE ADDRESS
264 000480* 012767 002446* 002220 MOV #RCVAF0,VA ;GET THE PHYSICAL ADDRESS OF FIRST
265 000482* 104415 000000* 002770* GETPAS,BEGIN,VA ;RECEIVER BUFFER
266 000484* 016767 002210 002174* MOV #PA,RCVADR ;GET PHYSICAL ADDRESS FROM 16-BIT VA
267 000486* 012767 002422* 002176* MOV #SYNC,VA ;SAVE RECEIVE BUFFER PHYSICAL ADDRESS
268 000488* 104415 000000* 002770* GETPAS,BEGIN,VA ;GET THE PHYSICAL ADDRESS OF TRANSMITTER BUFFER
269 000490* 016767 002166 002154* MOV #PA,XMTADR ;GET PHYSICAL ADDRESS FROM 16-BIT VA
270 000492* 000241 CLC ;SAVE TRANSMITTER BUFFER PHYSICAL ADDRESS
271 000494* 006167 002160 RDL EA ;BE SURE CARRY BIT IS CLEAR BEFORE ROTATING BITS
272 000496* 006167 002160 RDL EA ;ALIGN THE EXTENDED ADDRESS BITS IN ORDER
273 000498* 006167 002160 RDL EA ;TO SET BITS 13 AND 14 OF THE REG/ERR
274 000500* 006200 1\$: ASR R0 ;REGISTER
275 000502* 103407 BCS 35 ;ISOLATE DEVICE SELECTION FLAG IN "CM" BIT
276 000504* 001536 BEQ INITIAL ;IF SELECTED, GO INITIATE TRANSFER
277 000506* 062701 000010 002124 2\$: ADD #10,R1 ;IF NO MORE SELECTED, GO SETUP DATA
278 000508* 062701 000020 002124 ADD #20,RCVADR ;UPDATE POINTER TO NEXT DEVICE ADDRESS
279 000510* 000767 BR 15 ;UPDATE POINTER TO NEXT RECEIVER BUFFER
280 000512* 105061 000005 3\$: CLR# CLR# 5(R1) ;PROCESS NEXT DEVICE
281 000514* 012702 000020 MOV #20,R2 ;CLEAR SECONDARY REGISTERS POINTER
282 000516* 000642* 000020 FROM REG/ERR REGISTER(BITS 0-11)
283 000518* 052761 010000 000004 4\$: BIS #BIT12,4(R1) ;SET COUNT TO 16. FOR SECONDARY REGISTER
284 000520* 142761 000140 000005 BIC# #140,5(R1) ;CLEARING
285 000522* 005061 000006 CLR 6(R1) ;ENABLE EXTENDED BITS WRITING(REG/ERR BIT 12)
286 000524* 105261 000005 INCR 5(R1) ;CLEAR BITS 16 AND 17 OF THE
287 000526* 005302 DEC R2 ;SECONDARY REGISTER BY CLEARING 13
288 000528* 001364 BNE 45 ;AND 14 OF THE REG/ERR REGISTER
289 000530* 012702 000020 MOV #20,R2 ;CLEAR A SECONDARY REGISTER
290 000532* 112761 000017 000001 5\$: MOV#B #17,1(R1) ;POINT TO NEXT SECONDARY REGISTER
291 000534* 001417 BEQ 75 ;REDUCE COUNT. ARE ALL SIXTEEN DONE?
292 000536* 112761 000017 000001 5\$: MOV#B #17,1(R1) ;IF YES, CLEAR CHARACTER DETECT REGCS.
293 000538* 001417 BEQ 75 ;IS -DB OPTION HERE?
294 000540* 112761 000017 000001 5\$: MOV#B #17,1(R1) ;IF NO, GO DO NEXT ONE
295 000542* 001417 BEQ 75 ;SET UP COUNTER TO CLEAR CHAR. DETECT REGCS.
296 000544* 112761 000017 000001 5\$: MOV#B #17,1(R1) ;IF YES, CLEAR CHARACTER DETECT REGCS.
297 000546* 001417 BEQ 75 ;IS -DB OPTION HERE?
298 000548* 112761 000017 000001 5\$: MOV#B #17,1(R1) ;IF NO, GO DO NEXT ONE
299 000550* 001417 BEQ 75 ;SET UP COUNTER TO CLEAR CHAR. DETECT REGCS.
300 000552* 112761 000030 000005 5\$: MOV#B #30,5(R1) ;IF YES, CLEAR CHARACTER DETECT REGCS.
301 000554* 005061 CLR 6(R1) ;IS -DB OPTION HERE?
302 000556* 005061 CLR 6(R1) ;IF NO, GO DO NEXT ONE
303 000558* 005061 CLR 6(R1) ;SET UP COUNTER TO CLEAR CHAR. DETECT REGCS.
304 000560* 005061 CLR 6(R1) ;IF YES, CLEAR CHARACTER DETECT REGCS.
305 000562* 005061 CLR 6(R1) ;IS -DB OPTION HERE?
306 000564* 005061 CLR 6(R1) ;IF NO, GO DO NEXT ONE
307 000566* 005061 CLR 6(R1) ;SET UP COUNTER TO CLEAR CHAR. DETECT REGCS.
308 000568* 005061 CLR 6(R1) ;IF YES, CLEAR CHARACTER DETECT REGCS.
309 000570* 005061 CLR 6(R1) ;IS -DB OPTION HERE?
310 000572* 005061 CLR 6(R1) ;IF NO, GO DO NEXT ONE
311 000574* 005061 CLR 6(R1) ;SET UP COUNTER TO CLEAR CHAR. DETECT REGCS.
312 000576* 005061 CLR 6(R1) ;IF YES, CLEAR CHARACTER DETECT REGCS.
313 000578* 005061 CLR 6(R1) ;IS -DB OPTION HERE?
314 000580* 005061 CLR 6(R1) ;IF NO, GO DO NEXT ONE
315 000582* 005061 CLR 6(R1) ;SET UP COUNTER TO CLEAR CHAR. DETECT REGCS.
316 000584* 005061 CLR 6(R1) ;IF YES, CLEAR CHARACTER DETECT REGCS.
317 000586* 005061 CLR 6(R1) ;IS -DB OPTION HERE?
318 000588* 005061 CLR 6(R1) ;IF NO, GO DO NEXT ONE
319 000590* 005061 CLR 6(R1) ;SET UP COUNTER TO CLEAR CHAR. DETECT REGCS.
320 000592* 005061 CLR 6(R1) ;IF YES, CLEAR CHARACTER DETECT REGCS.
321 000594* 005061 CLR 6(R1) ;IS -DB OPTION HERE?
322 000596* 005061 CLR 6(R1) ;IF NO, GO DO NEXT ONE
323 000598* 005061 CLR 6(R1) ;SET UP COUNTER TO CLEAR CHAR. DETECT REGCS.
324 000600* 005061 CLR 6(R1) ;IF YES, CLEAR CHARACTER DETECT REGCS.
325 000602* 005061 CLR 6(R1) ;IS -DB OPTION HERE?
326 000604* 005061 CLR 6(R1) ;IF NO, GO DO NEXT ONE
327 000606* 005061 CLR 6(R1) ;SET UP COUNTER TO CLEAR CHAR. DETECT REGCS.
328 000608* 005061 CLR 6(R1) ;IF YES, CLEAR CHARACTER DETECT REGCS.
329 000610* 005061 CLR 6(R1) ;IS -DB OPTION HERE?
330 000612* 005061 CLR 6(R1) ;IF NO, GO DO NEXT ONE
331 000614* 005061 CLR 6(R1) ;SET UP COUNTER TO CLEAR CHAR. DETECT REGCS.
332 000616* 005061 CLR 6(R1) ;IF YES, CLEAR CHARACTER DETECT REGCS.
333 000618* 005061 CLR 6(R1) ;IS -DB OPTION HERE?
334 000620* 005061 CLR 6(R1) ;IF NO, GO DO NEXT ONE
335 000622* 005061 CLR 6(R1) ;SET UP COUNTER TO CLEAR CHAR. DETECT REGCS.
336 000624* 005061 CLR 6(R1) ;IF YES, CLEAR CHARACTER DETECT REGCS.
337 000626* 005061 CLR 6(R1) ;IS -DB OPTION HERE?
338 000628* 005061 CLR 6(R1) ;IF NO, GO DO NEXT ONE
339 000630* 005061 CLR 6(R1) ;SET UP COUNTER TO CLEAR CHAR. DETECT REGCS.
340 000632* 005061 CLR 6(R1) ;IF YES, CLEAR CHARACTER DETECT REGCS.
341 000634* 005061 CLR 6(R1) ;IS -DB OPTION HERE?
342 000636* 005061 CLR 6(R1) ;IF NO, GO DO NEXT ONE
343 000638* 005061 CLR 6(R1) ;SET UP COUNTER TO CLEAR CHAR. DETECT REGCS.
344 000640* 005061 CLR 6(R1) ;IF YES, CLEAR CHARACTER DETECT REGCS.
345 000642* 005061 CLR 6(R1) ;IS -DB OPTION HERE?
346 000644* 005061 CLR 6(R1) ;IF NO, GO DO NEXT ONE
347 000646* 005061 CLR 6(R1) ;SET UP COUNTER TO CLEAR CHAR. DETECT REGCS.
348 000648* 005061 CLR 6(R1) ;IF YES, CLEAR CHARACTER DETECT REGCS.
349 000650* 005061 CLR 6(R1) ;IS -DB OPTION HERE?
350 000652* 005061 CLR 6(R1) ;IF NO, GO DO NEXT ONE
351 000654* 005061 CLR 6(R1) ;SET UP COUNTER TO CLEAR CHAR. DETECT REGCS.
352 000656* 005061 CLR 6(R1) ;IF YES, CLEAR CHARACTER DETECT REGCS.
353 000658* 005061 CLR 6(R1) ;IS -DB OPTION HERE?
354 000660* 005061 CLR 6(R1) ;IF NO, GO DO NEXT ONE
355 000662* 005061 CLR 6(R1) ;SET UP COUNTER TO CLEAR CHAR. DETECT REGCS.
356 000664* 005061 CLR 6(R1) ;IF YES, CLEAR CHARACTER DETECT REGCS.
357 000666* 005061 CLR 6(R1) ;IS -DB OPTION HERE?
358 000668* 005061 CLR 6(R1) ;IF NO, GO DO NEXT ONE
359 000670* 005061 CLR 6(R1) ;SET UP COUNTER TO CLEAR CHAR. DETECT REGCS.
360 000672* 005061 CLR 6(R1) ;IF YES, CLEAR CHARACTER DETECT REGCS.
361 000674* 005061 CLR 6(R1) ;IS -DB OPTION HERE?
362 000676* 005061 CLR 6(R1) ;IF NO, GO DO NEXT ONE
363 000678* 005061 CLR 6(R1) ;SET UP COUNTER TO CLEAR CHAR. DETECT REGCS.
364 000680* 005061 CLR 6(R1) ;IF YES, CLEAR CHARACTER DETECT REGCS.
365 000682* 005061 CLR 6(R1) ;IS -DB OPTION HERE?
366 000684* 005061 CLR 6(R1) ;IF NO, GO DO NEXT ONE
367 000686* 005061 CLR 6(R1) ;SET UP COUNTER TO CLEAR CHAR. DETECT REGCS.
368 000688* 005061 CLR 6(R1) ;IF YES, CLEAR CHARACTER DETECT REGCS.
369 000690* 005061 CLR 6(R1) ;IS -DB OPTION HERE?
370 000692* 005061 CLR 6(R1) ;IF NO, GO DO NEXT ONE
371 000694* 005061 CLR 6(R1) ;SET UP COUNTER TO CLEAR CHAR. DETECT REGCS.
372 000696* 005061 CLR 6(R1) ;IF YES, CLEAR CHARACTER DETECT REGCS.
373 000698* 005061 CLR 6(R1) ;IS -DB OPTION HERE?
374 000700* 005061 CLR 6(R1) ;IF NO, GO DO NEXT ONE
375 000702* 005061 CLR 6(R1) ;SET UP COUNTER TO CLEAR CHAR. DETECT REGCS.
376 000704* 005061 CLR 6(R1) ;IF YES, CLEAR CHARACTER DETECT REGCS.
377 000706* 005061 CLR 6(R1) ;IS -DB OPTION HERE?
378 000708* 005061 CLR 6(R1) ;IF NO, GO DO NEXT ONE
379 000710* 005061 CLR 6(R1) ;SET UP COUNTER TO CLEAR CHAR. DETECT REGCS.
380 000712* 005061 CLR 6(R1) ;IF YES, CLEAR CHARACTER DETECT REGCS.
381 000714* 005061 CLR 6(R1) ;IS -DB OPTION HERE?
382 000716* 005061 CLR 6(R1) ;IF NO, GO DO NEXT ONE
383 000718* 005061 CLR 6(R1) ;SET UP COUNTER TO CLEAR CHAR. DETECT REGCS.
384 000720* 005061 CLR 6(R1) ;IF YES, CLEAR CHARACTER DETECT REGCS.
385 000722* 005061 CLR 6(R1) ;IS -DB OPTION HERE?
386 000724* 005061 CLR 6(R1) ;IF NO, GO DO NEXT ONE
387 000726* 005061 CLR 6(R1) ;SET UP COUNTER TO CLEAR CHAR. DETECT REGCS.
388 000728* 005061 CLR 6(R1) ;IF YES, CLEAR CHARACTER DETECT REGCS.
389 000730* 005061 CLR 6(R1) ;IS -DB OPTION HERE?
390 000732* 005061 CLR 6(R1) ;IF NO, GO DO NEXT ONE
391 000734* 005061 CLR 6(R1) ;SET UP COUNTER TO CLEAR CHAR. DETECT REGCS.
392 000736* 005061 CLR 6(R1) ;IF YES, CLEAR CHARACTER DETECT REGCS.
393 000738* 005061 CLR 6(R1) ;IS -DB OPTION HERE?
394 000740* 005061 CLR 6(R1) ;IF NO, GO DO NEXT ONE
395 000742* 005061 CLR 6(R1) ;SET UP COUNTER TO CLEAR CHAR. DETECT REGCS.
396 000744* 005061 CLR 6(R1) ;IF YES, CLEAR CHARACTER DETECT REGCS.
397 000746* 005061 CLR 6(R1) ;IS -DB OPTION HERE?
398 000748* 005061 CLR 6(R1) ;IF NO, GO DO NEXT ONE
399 000750* 005061 CLR 6(R1) ;SET UP COUNTER TO CLEAR CHAR. DETECT REGCS.
400 000752* 005061 CLR 6(R1) ;IF YES, CLEAR CHARACTER DETECT REGCS.
401 000754* 005061 CLR 6(R1) ;IS -DB OPTION HERE?
402 000756* 005061 CLR 6(R1) ;IF NO, GO DO NEXT ONE
403 000758* 005061 CLR 6(R1) ;SET UP COUNTER TO CLEAR CHAR. DETECT REGCS.
404 000760* 005061 CLR 6(R1) ;IF YES, CLEAR CHARACTER DETECT REGCS.
405 000762* 005061 CLR 6(R1) ;IS -DB OPTION HERE?
406 000764* 005061 CLR 6(R1) ;IF NO, GO DO NEXT ONE
407 000766* 005061 CLR 6(R1) ;SET UP COUNTER TO CLEAR CHAR. DETECT REGCS.
408 000768* 005061 CLR 6(R1) ;IF YES, CLEAR CHARACTER DETECT REGCS.
409 000770* 005061 CLR 6(R1) ;IS -DB OPTION HERE?
410 000772* 005061 CLR 6(R1) ;IF NO, GO DO NEXT ONE
411 000774* 005061 CLR 6(R1) ;SET UP COUNTER TO CLEAR CHAR. DETECT REGCS.
412 000776* 005061 CLR 6(R1) ;IF YES, CLEAR CHARACTER DETECT REGCS.
413 000778* 005061 CLR 6(R1) ;IS -DB OPTION HERE?
414 000780* 005061 CLR 6(R1) ;IF NO, GO DO NEXT ONE
415 000782* 005061 CLR 6(R1) ;SET UP COUNTER TO CLEAR CHAR. DETECT REGCS.
416 000784* 005061 CLR 6(R1) ;IF YES, CLEAR CHARACTER DETECT REGCS.
417 000786* 005061 CLR 6(R1) ;IS -DB OPTION HERE?
418 000788* 005061 CLR 6(R1) ;IF NO, GO DO NEXT ONE
419 000790* 005061 CLR 6(R1) ;SET UP COUNTER TO CLEAR CHAR. DETECT REGCS.
420 000792* 005061 CLR 6(R1) ;IF YES, CLEAR CHARACTER DETECT REGCS.
421 000794* 005061 CLR 6(R1) ;IS -DB OPTION HERE?
422 000796* 005061 CLR 6(R1) ;IF NO, GO DO NEXT ONE
423 000798* 005061 CLR 6(R1) ;SET UP COUNTER TO CLEAR CHAR. DETECT REGCS.
424 000800* 005061 CLR 6(R1) ;IF YES, CLEAR CHARACTER DETECT REGCS.
425 000802* 005061 CLR 6(R1) ;IS -DB OPTION HERE?
426 000804* 005061 CLR 6(R1) ;IF NO, GO DO NEXT ONE
427 000806* 005061 CLR 6(R1) ;SET UP COUNTER TO CLEAR CHAR. DETECT REGCS.
428 000808* 005061 CLR 6(R1) ;IF YES, CLEAR CHARACTER DETECT REGCS.
429 000810* 005061 CLR 6(R1) ;IS -DB OPTION HERE?
430 000812* 005061 CLR 6(R1) ;IF NO, GO DO NEXT ONE
431 000814* 005061 CLR 6(R1) ;SET UP COUNTER TO CLEAR CHAR. DETECT REGCS.
432 000816* 005061 CLR 6(R1) ;IF YES, CLEAR CHARACTER DETECT REGCS.
433 000818* 005061 CLR 6(R1) ;IS -DB OPTION HERE?
434 000820* 005061 CLR 6(R1) ;IF NO, GO DO NEXT ONE
435 000822* 005061 CLR 6(R1) ;SET UP COUNTER TO CLEAR CHAR. DETECT REGCS.
436 000824* 005061 CLR 6(R1) ;IF YES, CLEAR CHARACTER DETECT REGCS.
437 000826* 005061 CLR 6(R1) ;IS -DB OPTION HERE?
438 000828* 005061 CLR 6(R1) ;IF NO, GO DO NEXT ONE
439 000830* 005061 CLR 6(R1) ;SET UP COUNTER TO CLEAR CHAR. DETECT REGCS.
440 000832* 005061 CLR 6(R1) ;IF YES, CLEAR CHARACTER DETECT REGCS.
441 000834* 005061 CLR 6(R1) ;IS -DB OPTION HERE?
442 000836* 005061 CLR 6(R1) ;IF NO, GO DO NEXT ONE
443 000838* 005061 CLR 6(R1) ;SET UP COUNTER TO CLEAR CHAR. DETECT REGCS.
444 000840* 005061 CLR 6(R1) ;IF YES, CLEAR CHARACTER DETECT REGCS.
445 000842* 005061 CLR 6(R1) ;IS -DB OPTION HERE?
446 000844* 005061 CLR 6(R1) ;IF NO, GO DO NEXT ONE
447 000846* 005061 CLR 6(R1) ;SET UP COUNTER TO CLEAR CHAR. DETECT REGCS.
448 000848* 005061 CLR 6(R1) ;IF YES, CLEAR CHARACTER DETECT REGCS.
449 000850* 005061 CLR 6(R1) ;IS -DB OPTION HERE?
450 000852* 005061 CLR 6(R1) ;IF NO, GO DO NEXT ONE
451 000854* 005061 CLR 6(R1) ;SET UP COUNTER TO CLEAR CHAR. DETECT REGCS.
452 000856* 005061 CLR 6(R1) ;IF YES, CLEAR CHARACTER DETECT REGCS.
453 000858* 005061 CLR 6(R1) ;IS -DB OPTION HERE?
454 000860* 005061 CLR 6(R1) ;IF NO, GO DO NEXT ONE
455 000862* 005061 CLR 6(R1) ;SET UP COUNTER TO CLEAR CHAR. DETECT REGCS.
456 000864* 005061 CLR 6(R1) ;IF YES, CLEAR CHARACTER DETECT REGCS.
457 000866* 005061 CLR 6(R1) ;IS -DB OPTION HERE?
458 000868* 005061 CLR 6(R1) ;IF NO, GO DO NEXT ONE
459 000870* 005061 CLR 6(R1) ;SET UP COUNTER TO CLEAR CHAR. DETECT REGCS.
460 000872* 005061 CLR 6(R1) ;IF YES, CLEAR CHARACTER DETECT REGCS.
461 000874* 005061 CLR 6(R1) ;IS -DB OPTION HERE?
462 000876* 005061 CLR 6(R1) ;IF NO, GO DO NEXT ONE
463 000878* 005061 CLR 6(R1) ;SET UP COUNTER TO CLEAR CHAR. DETECT REGCS.
464 000880* 005061 CLR 6(R1) ;IF YES, CLEAR CHARACTER DETECT REGCS.
465 000882* 005061 CLR 6(R1) ;IS -DB OPTION HERE?
466 000884* 005061 CLR 6(R1) ;IF NO, GO DO NEXT ONE
467 000886* 005061 CLR 6(R1) ;SET UP COUNTER TO CLEAR CHAR. DETECT REGCS.
468 000888* 005061 CLR 6(R1) ;IF YES, CLEAR CHARACTER DETECT REGCS.
469 000890* 005061 CLR 6(R1) ;IS -DB OPTION HERE?
470 000892* 005061 CLR 6(R1) ;IF NO, GO DO NEXT ONE
471 000894* 005061 CLR 6(R1) ;SET UP COUNTER TO CLEAR CHAR. DETECT REGCS.
472 000896* 005061 CLR 6(R1) ;IF YES, CLEAR CHARACTER DETECT REGCS.
473 000898* 005061 CLR 6(R1) ;IS -DB OPTION HERE?
474 000900* 005061 CLR 6(R1) ;IF NO, GO DO NEXT ONE
475 000902* 005061 CLR 6(R1) ;SET UP COUNTER TO CLEAR CHAR. DETECT REGCS.
476 000904* 005061 CLR 6(R1) ;IF YES, CLEAR CHARACTER DETECT REGCS.
477 000906* 005061 CLR 6(R1) ;IS -DB OPTION HERE?
478 000908* 005061 CLR 6(R1) ;IF NO, GO DO NEXT ONE
479 000910* 005061 CLR 6(R1) ;SET UP COUNTER TO CLEAR CHAR. DETECT REGCS.
480 000912* 005061 CLR 6(R1) ;IF YES, CLEAR CHARACTER DETECT REGCS.
481 000914* 005061 CLR 6(R1) ;IS -DB OPTION HERE?
482 000916* 005061 CLR 6(R1) ;IF NO, GO DO NEXT ONE
483 000918* 005061 CLR 6(R1) ;SET UP COUNTER TO CLEAR CHAR. DETECT REGCS.
484 000920* 005061 CLR 6(R1) ;IF YES, CLEAR CHARACTER DETECT REGCS.
485 000922* 005061 CLR 6(R1) ;IS -DB OPTION HERE?
486 000924* 005061 CLR 6(R1) ;IF NO, GO DO NEXT ONE
487 000926* 005061 CLR 6(R1) ;SET UP COUNTER TO CLEAR CHAR. DETECT REGCS.
488 000928* 005061 CLR 6(R1) ;IF YES, CLEAR CHARACTER DETECT REGCS.
489 000930* 005061 CLR 6(R1) ;IS -DB OPTION HERE?
490 000932* 005061 CLR 6(R1) ;IF NO, GO DO NEXT ONE
491 000934* 005061 CLR 6(R1) ;SET UP COUNTER TO CLEAR CHAR. DETECT REGCS.
492 000936* 005061 CLR 6(R1) ;IF YES, CLEAR CHARACTER DETECT REGCS.
493 000938* 005061 CLR 6(R1) ;IS -DB OPTION HERE?
494 000940* 005061 CLR 6(R1) ;IF NO, GO DO NEXT ONE
495 000942* 005061 CLR 6(R1) ;SET UP COUNTER TO CLEAR CHAR. DETECT REGCS.
496 000944* 005061 CLR 6(R1) ;IF YES, CLEAR CHARACTER DETECT REGCS.
497 000946* 005061 CLR 6(R1) ;IS -DB OPTION HERE?
498 000948* 005061 CLR 6(R1) ;IF NO, GO DO NEXT ONE
499 000950* 005061 CLR 6(R1) ;SET UP COUNTER TO CLEAR CHAR. DETECT REGCS.
500 000952* 005061 CLR 6(R1) ;IF YES, CLEAR CHARACTER DETECT REGCS.
501 000954* 005061 CLR 6(R1) ;IS -DB OPTION HERE?
502 000956* 005061 CLR 6(R1) ;IF NO, GO DO NEXT ONE
503 000958* 005061 CLR 6(R1) ;SET UP COUNTER TO CLEAR CHAR. DETECT REGCS.
504 000960* 005061 CLR 6(R1) ;IF YES, CLEAR CHARACTER DETECT REGCS.
505 000962* 005061 CLR 6(R1) ;IS -DB OPTION HERE?
506 000964* 005061 CLR 6(R1) ;IF NO, GO DO NEXT ONE
507 000966* 005061 CLR 6(R1) ;SET UP COUNTER TO CLEAR CHAR. DETECT REGCS.
508 000968* 005061 CLR 6(R1) ;IF YES, CLEAR CHARACTER DETECT REGCS.
509 000970* 005061 CLR 6(R1) ;IS -DB OPTION HERE?
510 000972* 005061 CLR 6(R1) ;IF NO, GO DO NEXT ONE
511 000974* 005061 CLR 6(R1) ;SET UP COUNTER TO CLEAR CHAR. DETECT REGCS.
512 000976* 005061 CLR 6(R1) ;IF YES, CLEAR CHARACTER DETECT REGCS.
513 000978* 005061 CLR 6(R1) ;IS -DB OPTION HERE?
514 000980* 005061 CLR 6(R1) ;IF NO, GO DO NEXT ONE
515 000982* 005061 CLR 6(R1) ;SET UP COUNTER TO CLEAR CHAR. DETECT REGCS.
516 000984* 005061 CLR 6(R1) ;IF YES, CLEAR CHARACTER DETECT REGCS.
517 000986* 005061 CLR 6(R1) ;IS -DB OPTION HERE?
518 000988* 005061 CLR 6(R1) ;IF NO, GO DO NEXT ONE
519 000990* 005061 CLR 6(R1) ;SET UP COUNTER TO CLEAR CHAR. DETECT REGCS.
520 000992* 005061 CLR 6(R1) ;IF YES, CLEAR CHARACTER DETECT REGCS.
521 000994* 005061 CLR 6(R1) ;IS -DB OPTION HERE?
522 000996* 005061 CLR 6(R1) ;IF NO, GO DO NEXT ONE
523 000998* 005061 CLR 6(R1) ;SET UP COUNTER TO CLEAR CHAR. DETECT REGCS.
524 001000* 005061 CLR 6(R1) ;IF YES, CLEAR CHARACTER DETECT REGCS.
525 001002* 005061 CLR 6(R1) ;IS -DB OPTION HERE?
526 001004* 005061 CLR 6(R1) ;IF NO, GO DO NEXT ONE
527 001006* 005061 CLR 6(R1) ;SET UP COUNTER TO CLEAR CHAR. DETECT REGCS.
528 001008* 005061 CLR 6(R1) ;IF YES, CLEAR CHARACTER DETECT REGCS.
529 001010* 005061 CLR 6(R1) ;IS -DB OPTION HERE?
530 001012* 005061 CLR 6(R1) ;IF NO, GO

```

302 000732* 112761 000034 000005      MOVB #34,5(R1)      ;SELECT SECONDARY REG. 14
303 000740* 005061 000006                CLR R1              ;ARE ALL REGISTERS CLEAR?
304 000744* 005303 000000                DEC R2              ;
305 000748* 005303 000000                BGT 75              ;
306 000750* 105361 000001                DECB 1(R1)          ;SELECT THE NEXT PAIR OF CHARACTER DETECT REGIST
307 000754* 003361 000000                BGT 55              ;CLEAR 16 PAIRS OF REGISTERS
308 000756* 112761 000012 000005 75:    MOVB #12,5(R1)     ;HIGH BYTE OF RECEIVER CSR
309 000764* 052761 000040 000006        BIS #BIT5,6(R1)    ;SELECT SECONDARY REGISTER 12
310 000772* 112761 000020 000005        MOVB #20,5(R1)    ;SELECT MISCELLANEOUS REGISTER
311 001000* 156761 001770 000005        BISB EA,5(R1)     ;MASTER CLEAR THE D011 BY
312 001004* 015761 001746 000006        MOV RC,ADR,6(R1)  ;SETTING BITS 5 OF THE MISC REGISTER
313 001014* 105261 000005 000006        INCB 5(R1)        ;ENABLE EXTENDED MEMORY BITS
314 001020* 012761 177751 000006        MOV #15,6(R1)    ;AND SELECT THE RECEIVER BUSS
315 001024* 012761 000022 000005        MOV #22,5(R1)    ;ADDRESS REGISTER (SEC. REG.0)
316 001034* 156761 001734 000005        BISB EA,6(R1)    ;SET THE EA BITS
317 001042* 016761 001714 000006        MOV XMTADR,6(R1) ;LOAD ADDRESS OF RECEIVER BUFFER
318 001050* 105261 000005 000006        INCB 5(R1)        ;LOAD ADDRESS OF TRANSMITTER BUFFER
319 001054* 012761 177755 000006        MOV #19,6(R1)    ;SELECT THE RECEIVER BYTE COUNT REGISTER
320 001062* 112761 000031 000005        MOV #31,5(R1)    ;(SECONDARY REG. 1)
321 001070* 016761 001326 000006        MOV SYNC,6(R1)   ;SET COUNT FOR 15 CHARACTERS
322 001076* 105261 000005 000006        INCB 5(R1)        ;SELECT THE TRANSMITTER BUSS ADDRESS REGISTER
323 001102* 012761 004010 000006        MOV #4010,6(R1) ;SET THE EA BITS
324 001110* 052711 000042                BIS #42,(R1)      ;(SECONDARY REGISTER 2)
325 001114* 000642                BR 2$              ;LOAD ADDRESS OF TRANSMITTER BUFFER
326 001116* 012701 002426*                INITIAL:MOV #XMRTRF,R1 ;SELECT THE TRANSMITTER BYTE COUNT REGISTER
327 001122* 012702 000017                MOV #15,R2        ;(SECONDARY REGISTER 3)
328 001124* 127767 000026 001643 1$:    CMPB #26,DATA    ;LOAD TRANSMISSION BYTE COUNT
329 001134* 001009 001635                BNE 4$            ;SELECT THE SYNC CHARACTER REGISTER
330 001142* 105267 001631 5$:    MOVB DATA,(R1)+ ;(SECONDARY REGISTER 1)
331 001146* 105267 001625                INCB DATA        ;LOAD THE SYNC REGISTER WITH
332 001152* 005302 001625                DEC R2            ;TWO COPIES OF THE SYNC CHARACTER
333 001156* 016700 001574                MOV #4010,6(R1) ;SELECT THE MISC REGISTER (REG. 12)
334 001162* 016701 176620                MOV ADDR,R1      ;ENABLE EXTENDED MEMORY BITS AND
335 001166* 003404 2$:    RSR R0           ;AND STRIP SYNC(BIT 1)
336 001172* 001410 3$:    BEQ TMRSET       ;ADJUST POINTERS FOR NEXT DEVICE
337 001174* 062701 3$:    ADD #10,R1
338 001200* 000772 3$:    BR 2$            ;POINT R1 TO BEGINNING OF TRANSMIT TEXT
339 001202* 005211 4$:    INC (R1)         ;USE R2 AS A LOOP COUNTER
340 001204* 012761 000051 000002        MOV #51,2(R1)    ;IS THE DATUM THE SYNC CHARACTER?
341 001212* 000770                BR 3$            ;IF NO, GO MOVE IT INTO THE TRANSMITTER BUFFER
342 001214* 012767 000005 001544 TMRSET: MOV #5,TMRCNT ;USING DATA, BUILD A BINARY TEST PATTERN
343 001222* 005004 35:    TIMER: CLR R4    ;IF YES, SKIP IT
344 001224* 104407 000000*                BREAKS,BEGIN    ;SET DATA TO NEXT CHARACTER
345 001228* 104407 000000*                BREAKS,BEGIN    ;REDUCE COUNT, HAVE 15 CHARACTERS BEEN MADE?
346 001234* 105767 001536                TSTB DONFLG     ;IF NO, CONTINUE BUILDING TEXT
347 001240* 001433                BNE DONFLG     ;IF YES, BEGIN TRANSFER
348 001244* 005304                BEQ FINISH     ;LOAD DEVICE SELECTION PARAMETER INTO R0
349 001248* 001367                DEC R4         ;LOAD BASE ADDRESS INTO R1
350 001254* 005367 001514                DEC R4         ;ISOLATE A LINE SELECTION FLAG IN "C" BIT
351 001258* 001363 001516                MOVB DONFLG,R3 ;IF SELECTED, GO START DEVICE
352 001260* 040367 001472                BIC R3,SELECT  ;IF NO MORE SELECTED, GO START WATCHDOG TIMER
353 001264* 006003 1$:    ROR R3         ;UPDATE ADDRESS POINTER
354 001266* 103402                RCS 2$         ;GO START NEXT DEVICE
355 001270* 005204                INC R4         ;START RECEIVER (BIT 0=GO BIT)
356 001272* 000774 2$:    BR R4,NUMBA1  ;ENABLE TRANSMITTER(BIT 0) AND ERROR INTER-
357 001274* 010467 001104                MOV #4,NUMBA1   ;RUPT(BIT 3) ANDTRANSMITTER INTER-
358 001300* 104420 000000* 002404*                OTOAS,BEGIN,NUMBA1,M1 ;RUP(BIT 5)
359 001306* 002367*                ;GO ADJUST POINTER TO NEXT DEVICE
360 001310* 104403 000000* 002354*                MSGNS,BEGIN,HUNG ;LOAD THE TIMER COUNTING FACTOR
361 001316* 005767 01434                TST SELECT     ;USING R4, RETURN TO MONITOR 65536 TIMES
362 001322* 001002                BNE FINISH     ;TEMPORARY RETURN TO MONITOR...
363 001324* 104410 000000*                ENDS,BEGIN     ;THEN CONTINUE AT NEXT INSTRUCTION
364 001330* 104413 000000*                FINISH:        ;IF DONFLG IS CLEAR, EACH SELECTED DEVICE WAS
365 001334* 000167 177104                JMP SETUP2     ;SUCCESSFUL
366 001340* 010577 001042                XMTINT: MOV R5, #XMTQPI ;IF SO, PERFORM ENDPASS HOUSEKEEPING
367 001344* 000774                ;MONITOR SHALL TEST END OF PASS
368 001348* 000774                ;START NEXT TRANSFER
369 001350* 000774                ;
370 001354* 000774                ;
371 001358* 000774                ;
372 001362* 000774                ;
373 001366* 000774                ;
374 001370* 000774                ;
375 001374* 000774                ;
376 001378* 000774                ;
377 001382* 000774                ;
378 001386* 000774                ;
379 001390* 000774                ;
380 001394* 000774                ;
381 001398* 000774                ;
382 001402* 000774                ;
383 001406* 000774                ;
384 001410* 000774                ;
385 001414* 000774                ;
386 001418* 000774                ;
387 001422* 000774                ;
388 001426* 000774                ;
389 001430* 000774                ;
390 001434* 000774                ;
391 001438* 000774                ;
392 001442* 000774                ;
393 001446* 000774                ;
394 001450* 000774                ;
395 001454* 000774                ;
396 001458* 000774                ;
397 001462* 000774                ;
398 001466* 000774                ;
399 001470* 000774                ;
400 001474* 000774                ;
401 001478* 000774                ;
402 001482* 000774                ;
403 001486* 000774                ;
404 001490* 000774                ;
405 001494* 000774                ;
406 001498* 000774                ;
407 001502* 000774                ;
408 001506* 000774                ;
409 001510* 000774                ;
410 001514* 000774                ;
411 001518* 000774                ;
412 001522* 000774                ;
413 001526* 000774                ;

```

```

358 001202* 005211 4$:    INC (R1)         ;START RECEIVER (BIT 0=GO BIT)
359 001204* 012761 000051 000002        MOV #51,2(R1)    ;ENABLE TRANSMITTER(BIT 0) AND ERROR INTER-
360 001212* 000770                BR 3$            ;RUPT(BIT 3) ANDTRANSMITTER INTER-
361 001214* 012767 000005 001544 TMRSET: MOV #5,TMRCNT ;RUP(BIT 5)
362 001222* 005004 35:    TIMER: CLR R4    ;GO ADJUST POINTER TO NEXT DEVICE
363 001224* 104407 000000*                BREAKS,BEGIN    ;LOAD THE TIMER COUNTING FACTOR
364 001228* 104407 000000*                BREAKS,BEGIN    ;USING R4, RETURN TO MONITOR 65536 TIMES
365 001234* 105767 001536                TSTB DONFLG     ;TEMPORARY RETURN TO MONITOR...
366 001240* 001433                BNE DONFLG     ;THEN CONTINUE AT NEXT INSTRUCTION
367 001244* 005304                BEQ FINISH     ;IF DONFLG IS CLEAR, EACH SELECTED DEVICE WAS
368 001248* 001367                DEC R4         ;SUCCESSFUL
369 001254* 005367 001514                DEC R4         ;IF SO, PERFORM ENDPASS HOUSEKEEPING
370 001258* 001363 001516                MOVB DONFLG,R3 ;IF NOT, REDUCE COUNT AND BREAK AGAIN
371 001260* 040367 001472                BIC R3,SELECT  ;BREAK IF COUNT NOT EXCEEDED
372 001264* 006003 1$:    ROR R3         ;REDUCE TIMING FACTOR
373 001266* 103402                RCS 2$         ;BREAK AGAIN IF NO TIMEOUT
374 001270* 005204                INC R4         ;IF TIMEOUT OCCURRED, SAVE PRESENT FLAGS
375 001272* 000774 2$:    BR R4,NUMBA1  ;IN R3
376 001274* 010467 001104                MOV #4,NUMBA1  ;USE R3 TO DEACTIVATE HUNG DEVICE
377 001300* 104420 000000* 002404*                OTOAS,BEGIN,NUMBA1,M1 ;BY CLEARING ACTIVE SELECTION FLAG FROM
378 001306* 002367*                ;STORE AT M1 ;DEVICE SELECTION PARAMETER
379 001310* 104403 000000* 002354*                MSGNS,BEGIN,HUNG ;DETERMINE WHICH LINE WAS
380 001316* 005767 01434                TST SELECT     ;BAD FOR REPORTING PURPOSES
381 001322* 001002                BNE FINISH     ;IF THIS IS THE LINE, R4 CONTAINS CORRECT
382 001324* 104410 000000*                ENDS,BEGIN     ;LINE NUMBER...GO REPORT IT
383 001330* 104413 000000*                FINISH:        ;IF NOT, INCREMENT R4 WHICH WAS INITIALLY
384 001334* 000167 177104                JMP SETUP2     ;0 FROM THE PREVIOUS LOOP
385 001340* 010577 001042                XMTINT: MOV R5, #XMTQPI ;LOAD THE OFFSET TO THE CSR INTO TRANSMITTER QUE
386 001344* 000774                ;
387 001348* 000774                ;
388 001350* 000774                ;
389 001354* 000774                ;
390 001358* 000774                ;
391 001362* 000774                ;
392 001366* 000774                ;
393 001370* 000774                ;
394 001374* 000774                ;
395 001378* 000774                ;
396 001382* 000774                ;
397 001386* 000774                ;
398 001390* 000774                ;
399 001394* 000774                ;
400 001398* 000774                ;
401 001402* 000774                ;
402 001406* 000774                ;
403 001410* 000774                ;
404 001414* 000774                ;
405 001418* 000774                ;
406 001422* 000774                ;
407 001426* 000774                ;
408 001430* 000774                ;
409 001434* 000774                ;
410 001438* 000774                ;
411 001442* 000774                ;
412 001446* 000774                ;
413 001450* 000774                ;

```

```

414 001344* 062767 000002 001034 ADD #2,XMTQPI ;UPDATE THE QUEUE ENTRY POINTER
415 001352* 022767 002666* 001026 CMP #XMTQOE+20,XMTQPI ;HAS THE QUEUE BOUNDARY BEEN EXCEEDED?
416 001360* 001003 RNE IS ;IF IT HAS NOT EXCEEDED THE BOUNDARY,
417 ;GO DEFER SERVICE TO LEVEL 0
418 001362* 012767 002646* 001016 MOV #XMTQOE,XMTQPI ;IF IT HAS EXCEEDED THE BOUNDARY
419 001370* 012605 1$: MOV (SP),R5 ;RESTORE THE PREVIOUS R5 VALUE FROM STACK
420 ;-----
421 001372* 000004 000000* 001400* BIRQS,BEGIN,XMTSRV ; QUEUE UP TO CONTINUE AT XMTSRV AND RTI
422 ;-----
423 001400* 017701 001004 XMTSRV: MOV #XMTQOE,R1 ;FETCH THE OFFSET FROM THE QUEUE
424 001404* 062767 000002 000776 ADD #2,XMTQOE ;UPDATE THE QUEUE RETRIEVAL POINTER
425 001412* 022767 002666* 000770 CMP #XMTQOE+20,XMTQOE ;HAS THE QUEUE BOUNDARY BEEN EXCEEDED?
426 RNE IS ;IF NOT, CONTINUE PROCESSING
427 001422* 012767 002646* 000760 MOV #XMTQOE,XMTQOE ;IF YES, RESET POINTER TO BEGINNING OF QUEUE
428 001430* 011100 1$: MOV (R1),R0 ;LOAD THE CSR ADDRESS INTO R0
429 001432* 005760 000002 TST 2(R0) ;IF THERE WAS AN ERROR, BIT 15 OF REG/ERR IS SET
430 001436* 002017 BGE 2$ ;IF NO ERROR, PROCESS THE INTERRUPT
431 001440* 010067 176434 MOV R0,CSRA ;ON ERROR, LOAD THE DEVICE ADDRESS INTO CSRA
432 001444* 011067 176432 MOV (R0),ACSR ;LOAD THE CONTENTS INTO ACSR
433 001450* 016067 000002 176426 MOV 2(R0),ASTAT ;LOAD THE REGISTER/ERRCP REG. INTO STATC
434 001456* 005067 176424 CLR ERRTYP ;UNKNOWN ERROR
435 ;*****
436 001462* 104405 000000* 000000 HRDERS,BEGIN,NULL ;ERROR FLAG SET-CONTENTS OF REG/ERR IN STATC
437 ;*****
438 001470* 005004 CLR R4 ;COMPENSATE FOR PRINT DELAY ON TIMER
439 001472* 105067 001300 CLR# DNFLG
440 001476* 042710 000001* 2$: BIC #BIT0,(R0) ;DISABLE TRANSMITTER
441 001502* 104400 000000* EXIT$,BEGIN ;EXIT TO MONITOR, MODULE WAIT FOR INTERRUPT.
442 ;-----
443
444
445 001506* 010577 000700 RCVINT: MOV R5,RCVQPI ;ENTER OFFSET INTO RECEIVER QUEUE
446 001512* 012167 000002 000672 ADD #RCVQPI ;UPDATE THE RECEIVER ENTRY POINTER
447 001520* 022767 002706* 000664 CMP #RCVQOE+20,RCVQPI ;HAS THE QUEUE BOUNDARY BEEN EXCEEDED?
448 RNE IS ;IF NOT, CONTINUE PROCESSING
449 001526* 001003 1$: MOV #RCVQOE,RCVQPI ;IF SO, POINT POINTER TO QUEUE BEGINNING
450 001530* 012767 002666* 000654 MOV (SP),R5 ;RESTORE PREVIOUS VALUE OF R5
451 ;-----
452 001540* 000004 000000* 001546* BIRQS,BEGIN,RCVSRV ; QUEUE UP TO CONTINUE AT RCVSRV AND RTI
453 ;-----
454
455 001546* 017700 000642 RCVSRV: MOV #RCVQOE,R0 ;RETRIEVE OFFSET FROM RECEIVER QUEUE
456 001552* 062767 000002 000634 ADD #2,RCVQOE ;UPDATE THE QUEUE POINTER
457 001560* 022767 002706* 000626 CMP #RCVQOE+20,RCVQOE ;HAS THE QUEUE BOUNDARY BEEN EXCEEDED?
458 RNE IS ;IF NOT, CONTINUE PROCESSING
459 001566* 001003 1$: MOV #RCVQOE,RCVQOE ;IF SO, POINT POINTER TO QUEUE BEGINNING
460 001570* 012767 002666* 000616 MOV (R0),R1 ;PLACE CSR ADDRESS IN R1
461 001600* 005761 000004 TST 4(R1) ;WERE THERE ANY ERRORS (BIT 15 OF REG/ERR REG)?
462 001604* 002014 BGE 2$ ;IF NOT, CONTINUE PROCESSING
463 001612* 011167 176266 MOV R1,CSRA ;IF SO, LOAD BASE ADDRESS INTO CSRA
464 001616* 011167 176264 MOV (R1),ACSR ;MOVE CONTENTS OF CSR TO ACSR
465 001624* 005067 176256 CLR 4(R1),ASTAT ;LOAD REG/ERR REG. CONTENTS INTO STATC
466 ERRTYP ;UNKNOWN ERROR
467 ;*****
468 001630* 104405 000000* 000000 HRDERS,BEGIN,NULL ;DQ11 ERROR FLAG SET-REG/ERR IN STATC
469 ;*****
470 001636* 122711 000246 2$: CMPB #246,(R1) ;

```

```

470 001642* 001412 BEQ 3$
471 001644* 010167 176230 MOV R1,CSRA
472 001650* 011167 176226 MOV (R1),ACSR
473 001654* 012767 000017 176224 MOV #1,ERRTYP ;UNKNOWN RECEIVER ERROR
474 ;*****
475 001662* 104405 000000* 000000 HRDERS,BEGIN,NULL ;RECEIVER STATUS ERROR
476 ;*****
477 001670* 122761 000254 000002 3$: CMPB #254,(R1) ;
478 001676* 001416 BEQ 4$ ;IF NO ERRORS,BEGIN COMPARING DATA
479 001700* 016167 000002 176174 MOV 2(R1),ACSR ;LOAD TRANSMITTER STATUS INTO ACSR
480 001706* 010167 176166 MOV R1,CSRA ;LOAD STATUS REGISTER ADDRESS
481 001712* 062767 000002 176160 ADD #2,CSRA ;INCR BASE ADDRESS TO SHOW TRANSMITTER ADDRESS
482 001720* 012767 000020 176160 MOV #2,ERRTYP ;UNKNOWN XMITTER ERROR
483 ;*****
484 001726* 104405 000000* 000000 HRDERS,BEGIN,NULL ;TRANSMITTER STATUS ERROR
485 ;*****
486
487
488
489 001734* 012767 002446* 001016 CKDATA: MOV #RCVQOE,RCVADR ;RESTORE BASE VALUE OF RECEIVER BUFFERS
490 001742* 012767 000017 001014 MOV R15,DATACT ;LOAD THE NUMBER OF CHARACTERS TO BE CHECKED
491 001750* 016002 000002 MOV 2(R0),R2 ;LOAD THE LINE NUMBER OF THIS DQ11
492 001754* 001405 1$: BEQ 2$ ;WHEN R2 IS 0, THE CORRECT RECEIVER BUFFER
493 ;HAS BEEN SELECTED
494 001756* 062767 000020 000774 ADD #20,RCVADR ;IF R2 IS NOT 0, POINT RCVADR TO NEXT BUFFER
495 001764* 005302 DEC R2 ;REDUCE THE LINE NUMBER BY 1
496 001766* 000772 BR 1$ ;GO SEE IF THE CORRECT BUFFER HAS BEEN DETERMINED
497 001770* 016702 2$: MOV RCVADR,R2 ;LOAD R2 WITH THE START OF THE RECEIVER BUFFER
498 001774* 012767 002446* 000756 MOV #RCVQOE,RCVADR ;RESTORE BASE VALUE OF RECEIVER BUFFERS
499 002002* 012703 002426* MOV #XMTBUF,R3 ;LOAD R3 WITH THE START OF TRANSMITTER TEXT
500 002006* 122223 3$: CMPB (R2)*,(R3)+ ;WATCH CHARACTERS, ARE THEY THE SAME?
501 002010* 001453 BEQ 4$ ;IF YES, GO PROCESS NEXT CHARACTER
502 002012* 010167 176062 MOV R1,CSRA ;LOAD THE BASE ADDRESS INTO CSRA
503 002016* 011167 176060 MOV (R1),ACSR ;LOAD RECEIVER CSR CONTENTS INTO ACSR
504 002022* 114267 176062 MOV#B -(R2),AWAS ;LOAD THE ACTUAL RECEIVED CHARACTER
505 002026* 114367 176054 MOV#B -(R3),ASB ;LOAD THE TRANSMITTED CHARACTER
506 002030* 010267 176046 MOV R2,WASADR ;LOAD THE CHARACTER RECEIVED ADDRESS
507 002036* 010367 176040 MOV R3,SABDR ;LOAD THE CHARACTER TRANSMITTED ADDRESS
508 002042* 016705 000350 MOV ERQPI,R5 ;USING R5, LOAD THE PRESENT VALUES OF THE REGISTER
509 ;INTO THE ERROR QUEUE, UNQUEUEING THEM AFTER THE
510 ;DATA ERROR CALL, THIS ASSURES THAT THE REGISTER
511 ;CONTENTS ARE VALID
512 002046* 010025 MOV R0,(R5)+ ;SAVE THE QUEUE OFFSET
513 002050* 010125 MOV R1,(R5)+ ;SAVE THE CSR ADDRESS
514 002052* 010225 MOV R2,(R5)+ ;SAVE THE PRESENT RECEIVE BUFFER POINTER
515 002054* 010325 MOV R3,(R5)+ ;SAVE THE CURRENT TRANSMIT BUFFER POINTER
516 002056* 010425 MOV R4,(R5)+ ;SAVE THE TIMER COUNTDOWN
517 002060* 020527 002732* MOV #ERRQOE+20, ;HAS THE POINTER EXCEEDED THE QUEUE BOUNDARY?
518 002064* 103402 BLD 5$ ;IF SO, DO NOT RESET POINTER
519 002066* 012705 002706* MOV #ERRQOE,R5 ;POINT POINTER TO BEGINNING OF QUEUE
520 002072* 010567 000320 5$: MOV #5,ERRQPI ;RESTORE ERROR QOE ENTRY POINTER
521 ;*****
522 002076* 104404 000000* DATERS,BEGIN ;DATA ERROR!!!
523 ;*****
524 002102* 016705 000312 MOV ERROPI,R5 ;LOAD R5 WITH ERROR QUEUE RETRIEVAL POINTER
525 ;RETRIEVE, USING R5, THE FOLLOWING FROM THE ERROR

```

```

526      002106* 012500      MOV      (R5)+,R0      ;QUEUE, PLACING THEM IN THE CORRESPONDING REGISTE
527      002110* 012501      MOV      (R5)+,R1      ;THE OFFSET ADDRESS...
528      002111* 012502      MOV      (R5)+,R2      ;THE BASE ADDRESS...
529      002112* 012503      MOV      (R5)+,R3      ;THE CURRENT RECEIVER BUFFER POINTER...
530      002113* 012504      MOV      (R5)+,R4      ;THE CURRENT TRANSMITTER BUFFER POINTER...
531      002114* 012505      MOV      (R5)+,R5      ;AND THE CORRECT TIMER VALUE
532      002120* 020527 002732* CMP      R5,ERRRQUE+20. ;HAS THE POINTER EXCEEDED THE QUEUE BOUNDARY?
533      002121* 103402      BLD      6S          ;IF NOT, DO NOT RESET IT
534      002122* 012705 002706* MOV      ERRRQUE,R5    ;IF SO, RESET THE POINTER TO QUEUE BEGINNING
535      002123* 010567 000262* MOV      R5,ERRRQUE   ;RESTORE THE RETRIEVAL POINTER
536      002136* 122223      CMPB     (R2)+,(R3)+  ;AUTOMATICALLY THE BUFFER POINTERS TO
537      ;POINT TO THE CORRECT CHARACTER
538      002140* 005367 000620 4S:   DEC      DACTACT      ;REDUCE CHARACTER COUNT, ARE ALL 15 DONE?
539      002144* 003320      BGT      3$          ;IF NO, DO NEXT CHARACTER
540
541
542
543
544      002146* 005003      RCVDDN: CLR      R3          ;IF YES, SETUP R3 TO TURN OFF FLAG IN DONFLG
545      ;FOR THIS LINE
546      002150* 016002 000002 1S:   MOV      2(R0),R2     ;LOAD THE LINE NUMBER INTO R2
547      002151* 000261      SEC          ;USING THE CARRY BIT, CREATE A ONE-BIT MASK
548      002152* 006103      RDL      R3          ;POINT THE MASK TO THE NEXT LINE
549      002160* 005302      DEC      R2          ;REDUCE THE LINE NUMBER
550      002164* 140367      BGE      1$          ;IF POSITIVE OR 0, GO SHIFT BIT AGAIN
551      ;IF NEGATIVE, THE MASK IS CORRECTLY ALIGNED-
552      002170* 104400 000000* EXITS,BEGIN ;CLEAR THE DONE BIT FOR THIS LINE
553      ;EXIT TO MONITOR. MODULE WAIT FOR INTERRUPT.
554      002174* 004567 177306 LNKTAB: JSR      R5,RCVINT ;LINK FOR RECEIVER 0
555      002200* 000000      0
556      002200* 000000      0
557      002204* 004567 177130      JSR      R5,XMTINT    ;LINK FOR TRANSMITTER 0
558      002210* 000000      0
559      002210* 000000 177270      JSR      R5,RCVINT    ;LINK FOR RECEIVER 1
560      002216* 000000      1
561      002220* 000001      1
562      002222* 004567 177112      JSR      R5,XMTINT    ;LINK FOR TRANSMITTER 1
563      002226* 000000      0
564      002230* 004567 177252      JSR      R5,RCVINT    ;LINK FOR RECEIVER 2
565      002234* 000000      0
566      002236* 000002      2
567      002240* 004567 177074      JSR      R5,XMTINT    ;LINK FOR TRANSMITTER 2
568      002244* 000000      0
569      002246* 004567 177234      JSR      R5,RCVINT    ;LINK FOR RECEIVER 3
570      002252* 000000      0
571      002254* 000003      3
572      002256* 004567 177056      JSR      R5,XMTINT    ;LINK FOR TRANSMITTER 3
573      002262* 000000      0
574      002264* 004567 177216      JSR      R5,RCVINT    ;LINK FOR RECEIVER 4
575      002270* 000000      0
576      002272* 000004      4
577      002274* 004567 177040      JSR      R5,XMTINT    ;LINK FOR TRANSMITTER 4
578      002300* 000000      0
579      002306* 000000 177200      JSR      R5,RCVINT    ;LINK FOR RECEIVER 5
580      002306* 000000      0
581      002310* 000005      5

```

```

582      002312* 004567 177022      JSR      R5,XMTINT    ;LINK FOR TRANSMITTER 5
583      002316* 000000      0
584      002320* 004567 177162      JSR      R5,RCVINT    ;LINK FOR RECEIVER 6
585      002324* 000000      0
586      002326* 000006      6
587      002330* 000000 177004      JSR      R5,XMTINT    ;LINK FOR TRANSMITTER 6
588      002334* 000000      0
589      002336* 004567 177144      JSR      R5,RCVINT    ;LINK FOR RECEIVER 7
590      002342* 000000      0
591      002344* 000007      7
592      002346* 004567 176766      JSR      R5,XMTINT    ;LINK FOR TRANSMITTER 7
593      002352* 000000      0
594
595      002354* 002360* HUNG: MESSAG
596      002356* 177777
597      002360* 042504 044526 042503 MESSAG: .ASCII "DEVICE "
598      002369* 040
599      002367* 000006
600      002375* 040 052510 043516 M1: .BLKB 6 HUNG
601      002402* 000
602      002404* 002404* .EVEN
603
604      002404* 000000 NUMBA1: .WORD 0
605      002406* 000000 XMTQPI: .WORD OPEN
606      002410* 000000 XMTQPO: .WORD OPEN
607      002412* 000000 RCVQPI: .WORD OPEN
608      002414* 000000 RCVQPO: .WORD OPEN
609      002416* 000000 ERRQPI: .WORD OPEN
610      002420* 000000 ERRQPO: .WORD OPEN
611
612      002422* 026 026 SYNC: .BYTE 26,26
613      002424* 026 026 .BYTE 26,26
614      002426* 000010 XMTBUF: .BLKW 8. ;TRANSMITTER CHARACTER BUFFER
615      002428* 000010 RCVBF0: .BLKW 8. ;DEVICE 0 RECEIVER BUFFER
616      002466* 000010 RCVBF1: .BLKW 8. ;DEVICE 1 RECEIVER BUFFER
617      002506* 000010 RCVBF2: .BLKW 8. ;DEVICE 2 RECEIVER BUFFER
618      002526* 000010 RCVBF3: .BLKW 8. ;DEVICE 3 RECEIVER BUFFER
619      002546* 000010 RCVBF4: .BLKW 8. ;DEVICE 4 RECEIVER BUFFER
620      002566* 000010 RCVBF5: .BLKW 8. ;DEVICE 5 RECEIVER BUFFER
621      002606* 000010 RCVBF6: .BLKW 8. ;DEVICE 6 RECEIVER BUFFER
622      002626* 000010 RCVBF7: .BLKW 8. ;DEVICE 7 RECEIVER BUFFER
623
624
625      002646* 000010 XMTQUE: .BLKW 8. ;TRANSMITTER SERVICE QUEUE
626      002666* 000010 RCVQUE: .BLKW 8. ;RECEIVER SERVICE QUEUE
627      002706* 000024 ERRQUE: .BLKW 20. ;ERROR SERVICE QUEUE
628
629
630      002756* 000000 SELECT: OPEN ;DEVICE SELECTION PARAMETER
631      002760* 000000 RCVADR: OPEN ;PHYSICAL ADDRESS OF RECEIVER BUFFERS
632      002762* 000000 XMTADR: OPEN ;PHYSICAL ADDRESS OF TRANSMITTER BUFFER
633      ;FOR ALL DEVICES
634      002764* 000000 DACTACT: .WORD OPEN ;COUNTER FOR NUMBER OF DATA ITEMS
635      002766* 000000 THRCNT: .WORD OPEN ;LOCATION FOR WATCHDOG TIMER FACTOR
636      002770* 000000 VA: .WORD OPEN ;LOCATION OF VIRTUAL ADDRESS PARAMETER
637      002772* 000000 PA: .WORD OPEN ;LOCATION OF PHYSICAL ADDRESS PARAMETER

```


DQAI DEC/X11 SYSTEM EXERCISER MODULE MACY11 30A(1052) 12-OCT-78 16:30 PAGE 20
XDQAI0.P11 12-OCT-78 11:54 CROSS REFERENCE TABLE -- USER SYMBOLS

SEC 001F

XDQAI0, XDQAI0/SDL/CRF:SYM=DDXCOM, XDQAI0
RUN-TIME: 1 2 .3 SECONDS
RUN-TIME RATIO: 15/4=3.2
CORE USED: 7K (13 PAGES)